REINFORCED CO-LEARNING FOR SEMI-SUPERVISED RANKING

BY

SHIBI HE

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2018

Urbana, Illinois

Adviser:

Professor Jian Peng

ABSTRACT

Learning to rank is vital to information retrieval and recommendation systems. Directly optimizing the listwise evaluation measure such as normalized discounted cumulative gain (NDCG) is an advanced way to learn a ranking model. However, this is only suited for training data with effective labels. In real applications, we are more often faced with the semi-supervised setting that only a partial set of data has labels. In this paper, we propose a co-learning strategy for the semi-supervised ranking problem. Our model has two modules: the classifier module and the reinforcement ranker module. Given a query, the classifier module is trained to classify whether a document is relevant or not. The reinforcement ranker module is trained to give relevance scores on the basis of treating ranking problems as Markov decision processes (MDP). We name our approach "reinforced co-learning" because the two modules are iteratively optimized and affect each other while training. When training the classifier module, we use the reinforcement module to give every candidate a relevance score and sample lower scored documents as irrelevant samples (negative samples). Likewise, in order to train the reinforcement ranker module, we use the classifier module to predict labels in the sequence in order to calculate the combined rewards. The linkage between the two modules is also reflected in the network structure. We add the feature sharing layer, which enables the classifier to distill its intermediate representations to the learning of reinforcement ranker module. Extensive experiments and ablation studies show that both our co-learning strategy and feature sharing can improve semi-supervised ranking problems.

To my parents, for their love and support.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

Learning to rank is essential for various retrieval and recommendation systems. Given a query, the goal is to provide a ranked list of items, which include a set of documents in a recommendation system or a corpus of web pages in web search.

Of various strategies for learning-to-rank, an intuitive way is to optimize evaluation metrics directly. Since evaluation metrics are not always differentiable, MDPRank [1] seeks to optimize the Normalized discounted cumulative gain (NDCG) scores by policy gradient methods [2]. This ranking approach considers the construction of ranking set to be a sequential Markov decision process (MDP), each step corresponding to selecting an item into the list. MDPRank shows promising results in the supervised ranking task, where all training data need to have labels in order to calculate the intermediate rewards at each step.

However, given the difficulty of acquiring labels in most real cases, we are more often faced with a large amount of unlabeled data. It is more interesting to study the semi-supervised setting [3], where only a partial set of training data has labels. MDPRank [1] cannot be directly used in this semi-supervised setting given that we cannot calculate an intermediate NDCG score for unlabeled data during training. Therefore, the adaption of reinforcement training into this semi-supervised learning-to-rank problem is a challenging problem.

In parallel, IRGAN [4] proposes a unified model for information retrieval modeling with the adversarial training [5]. Its discriminator mines information from both labeled and unlabeled data, providing the guidance to train the generator. Its generator works toward attacking the discriminator with difficult examples. IRGAN makes use of unlabeled data when its generator needs to sample negative documents to train the discriminator. Therefore, IRGAN is a strong baseline and comparison method in this work.

The recent advances of supervised deep learning techniques [6] in computer vision, speech recognition and natural language processing have tremendously improved the performance on challenging tasks, including image processing [7], speech-based translation [8] and language modeling [9]. The core idea of deep learning is to use artificial neural networks to model complex hierarchical or compositional data abstractions and representations from raw input data [10]. However, we are still far from building intelligent solutions for many real-world challenges, such as autonomous driving, human-computer interaction and automated decision making, in which software agents need to consider interactions with a dynamic environment and take actions towards goals. Reinforcement learning [11, 12, 13, 14] studies these problems and algorithms which learn policies to make decisions so as to maximize a reward signal from the environment. One of the promising algorithms is Q-learning [15, 16].

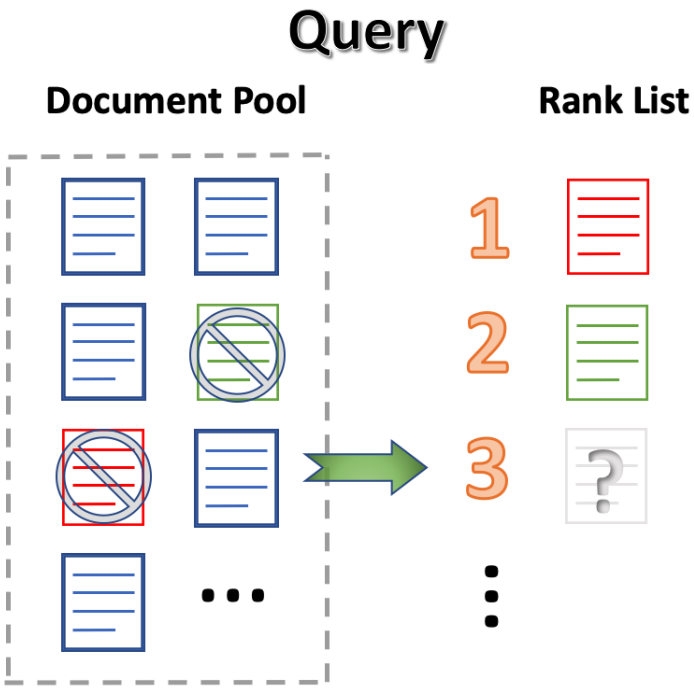

Figure 1.1: Ranking as an MDP

Deep reinforcement learning with neural function approximation [17, 18, 19, 20], possibly a first attempt to combine deep learning and reinforcement learning, has been proved to be effective on a few problems which classical AI approaches were unable to solve. Notable examples of deep reinforcement learning include human-level game playing [20] and AlphaGo [21].

In this paper, we propose a reinforcement learning method that can leverage the direct optimization of ranking scores and the learning of a large amount of unlabeled data at the same time. Like MDPRank [1], we formulate the process of document ranking as a Markov Decision Process in Figure (1.1). The constructions of a document ranking list would be considered as sequential decision makings, where each time step corresponds to a ranking position and each action is to select a document for its current position. Different from MDPRank, our model has two modules: *reinforcement ranker module* and *classifier module*. As shown in Figure (3.2), given the query, the classifier is trained to determine the predicted relevance of the retrieved document. In order to train the classifier, positive samples are chosen from the labeled set, and the negative data is sampled both from the labeled set and a distribution predicted by the reinforcement ranker module. The core module for ranking is the reinforcement ranker. Its input features have two parts. One part is from the original feature space, and the other is transferred from the classifier module. We consider that these intermediate features from the classifier can distill discriminative side

information [22, 23, 24]. The classifier module is not only affecting the input features by distillation, but also providing rewards for the training of the reinforcement ranker. We use the policy gradient method to train the reinforcement ranker. The reward is combined by two components: one is the NDCG score, which is used in MDPRank as well, and the other is the discriminative score output by the classifier module. This score reflects the accuracy with which the current retrieved list is related to the query.

We train two modules iteratively with the training algorithm that is displayed in Figure (3.1). In each step, the classifier would acquire more and more confident negative data, as the reinforcement ranker module becomes more and more accuracy; the reinforcement ranker module is updated by the policy gradient, given combined rewards and distilled features from the classifier module. Since both modules are learned together, we name our approach "reinforced co-learning" (co-learning).

Our contributions can be summarized as below:

- We propose a co-learning framework for semi-supervised ranking tasks. A classifier module and a reinforcement ranker module are designed for leveraging the unlabeled data and learning to rank. An iterative training pipeline is used to train our co-learning framework.

- A combined reward is designed to help the reinforcement ranker gain a better ranking performance.

- Knowledge distillation is deployed between the classifier and the reinforcement ranker. The classifier shares its intermediate representation as knowledge and passes it through to guide the training of the reinforcement ranker. This technique further improves the ranking performance.

The rest of the paper is organized as follows. We discuss related work in Chapter 2. We state the problem and formulate our approach in Chapter 3. Next, Sections 3.1, 3.2, and 3.3 explain two modules and the training pipeline in detail. After that, the experiments and ablation studies are conducted in Chapter 4. Finally, conclusions are given in Chapter 5.

CHAPTER 2: REVIEW OF PREVIOUS RESEARCH

2.1 LEARNING TO RANK

Learning to rank is a central problem in information retrieval. Many tasks could be naturally formulated as a ranking problem, such as web search [25], text retrieval [26], collaborative filtering [27], key term extraction [28], definition finding [29], sentiment analysis [30], and product rating [31]. In this section, we mainly focus on the document retrieval task where documents can be web pages, emails, academic papers, journals, books, and articles. Recently, as a huge amount of training data become available, machine learning tends to be an effective ranking model. "Learning-to-rank" refers to the methods that learn to leverage the predefined features by discriminative learning.

Document retrieval is a task as follows. The ranking system maintains a collection of documents. Given a query, the system retrieves documents which contain the query words from the collection. The system then ranks the documents, and returns the top ranked documents. The ranking task is conducted by using a ranking model $f(q, d)$ to sort the documents, where q and d denote a query and a document.

Traditionally, the ranking model $f(q, d)$ is created without training. For instance, in the BM25 model, based on the assumption, $f(q, d)$ is represented by a conditional probability distribution $P(r|q, d)$, where r takes 1 or 0 as value. 1 denotes relevant and 0 denotes irreverent. q and d denote a query and a document respectively. In Language Model for IR (LMIR) , $f(q, d)$ is represented as a conditional probability distribution $P(q|d)$. This probability model could be calculated with the words appearing in the queries and documents, and thus no training is needed.

A new trend has recently arisen in document retrieval area, particularly in web search, which is, to deploy machine learning methods to automatically construct the ranking model $f(q, d)$. The motivation of using machine learning techniques is based on a number of facts. Take web search as an example, there are various signal data that could represent relevance, such as the anchor texts and PageRank score of a web page. Thus, it is very natural to incorporate such information into the ranking model and automatically construct the ranking model using machine learning techniques. In web search engines, as a huge amount of search log data (click through data) is accumulated, the intuition of using the data driven technique (machine learning) becomes possible. One could use data from search log and automatically create the ranking model. As a matter of fact, learning to rank has become the most important technology for modern web search and ranking.

Various methods have been proposed to solve the ranking problem. Generally speaking, there are three approaches: pointwise approach, pairwise approach, and listwise approach. The input space of the pointwise approach is the feature vector and a single document. The output space is the relevance of each document. The accuracy is examined by the prediction of labels and ground truth labels for each document. Different loss functions could be used in pointwise approach such as classification, regression, and ordinal regression [32, 33, 34]. Some drawbacks of pointwise approach are that the position of a document in the final ranked list is invisible to the function and the algorithm intrinsically ignores the dependency of a group of documents given the same query. These problems limit the accuracy of pointwise approach.

The pairwise approach considers the preference between a pair of documents, for instance [25, 35, 36, 37]. The output space contains pairwise preference (which takes the value from -1, +1 between a pair of documents). The loss function in pairwise approach measures the inconsistency between the predicted and ground-truth labels. To be noticed that the loss function used in the pairwise approach only considers the relative order. When we focus on only a pair of documents, however, the position of the documents in the final list is hardly to be derived. Since the relative order in each pair of documents is the only consideration in loss function, determining the position of documents in the final ranked list could also be a bottleneck. Furthermore, the approach ignores the fact that some pairs are generated from the documents associated with the same query.

Some drawbacks of pointwise or pairwise approach are that the position of a document in the final ranked list is invisible to the function and the algorithm intrinsically ignores the dependency of a group of documents given the same query.

The listwise approach takes the entire group of documents given a query into consideration. Generally, two types of output are used, the relevance degree of all documents and permutation of the document list. Therefore, two types of loss functions are their counterparts. One loss function is to measure the labels of each document, and the other is to compare the ranked list with the ground truth list. Some listwise approaches include [38, 39, 40, 41, 42]. The biggest advantage of listwise approach compared with previous methods is that its loss function naturally considers the position of documents in the ranked list. The listwise approaches with a loss that defined on both document pairs and a list permutation weight added document pairs, e.g., LambdaRank [38] and LambdaMART [43], often achieve the best performance among various learning-to-rank algorithms.

Among these approaches, some are directly optimizing the evaluation measures. A number of approaches have been developed, such as SVMMAP [42], SVMNDCG [44], Adarank [45], and PermuRank [46]. This way of learning-to-rank has been proved to be very effective

in model training, parameter tuning, evidence combining, etc. In training, these methods construct a loss function that considers the prediction, ground truth and evaluation measures. The approximation of the loss function or a convex upper bound based on the loss is designed and optimized. Then at the ranking, the learned model assigns relevance scores to each document. SoftRank [40] approximates original measure score of normalized discounted cumulative gain (NDCG). SmoothRank [47] gives an approximated measure scores using the document position in the rank list. Other methods such as SVMMAP [42], SVMNDCG [44] and AdaRank [45] optimize a continuous and differentiable ranking error.

2.2 SEMI-SUPERVISED LEARNING, REINFORCEMENT LEARNING AND OTHER MACHINE LEARNING TECHNIQUES

A number of approaches have been proposed for semi-supervised learning. Bootstrapping is one of the most prevalent methods in semi-supervised learning [48, 3] including self-training [49, 50], co-training [51] and generative models [52, 53]. The Bootstrapping assumption is that predicted labels of unlabeled data could be used for supervised training.

Self-training first trains an initial classifier on a small amount of labeled data. It predicts labels for the unlabeled data, then adds confident predictions to the training set. Next, the classifier is re-trained and the whole process is repeated again. The effectiveness of self-training relies on the quality of predicted labels. Noisy labels would degrade the performance very much when added to the training set as classification mistake can reinforce itself. Self-training has been applied to natural language processing tasks [49, 54, 55], object detection [56], etc. Generative model with the EM algorithm [57] could be considered as a soft version self-training that models a joint distribution. The unlabeled data improves the accuracy when the model form is correctly designed.

Co-training [51] trains two classifiers on separated feature spaces from labeled data. Classifiers are used to predict labels of unlabeled data and further teach each other. Specifically, classifier A adds its confident predictions to the training set of classifier B and vise versa. Experiments and analysis [51, 58] show that co-training performs well when having high quality feature separations that are independent of each other given the class. [59, 60] applies co-training for information extraction and [61, 62] study the effectiveness when few labeled data is given.

To create a good data separation, instead of using predetermined rules, reinforce co-training [63] proposes to train a Q-learning as data selection policy, where rewards are given by classifier's accuracy on the validation set. Reinforced co-training is also related to the meta learning or "learning to learn" methods [64, 65, 66, 67] where a model is trained

to find optimal parameters for another model.

Our co-learning is different from the aforementioned methods in several aspects. Unlike co-training, our framework has only one classifier thus no feature split or data selection is needed. Self-training adds unlabeled data to the training set with the predicted labels, while our co-learning does not.

More generally, we can define the learning paradigm that utilizes the agreement among different learners. It is not required in multiview learning models that the particular assumptions of Co-Training hold. Instead, several hypotheses with different inductive basies such as decision trees and SVMs trained from the same labeled dataset are required to make the similar prediction on given unlabeled instances.

Multiview learning has a long history and it has been applied to semi-supervised regression [68, 69], and the more challenging structured output spaces [70]. Theoretical analysis has been conducted on the value of agreement among multiple learners which could be found in [71, 72].

Knowledge distillation is considered as an ensemble technique that retains the performance with a compressed model. The performance of knowledge transfer is very sensitive to the definition of distilled knowledge. The distilled knowledge could be extracted by various part of features in the pretrained deep neural network (DNN). One can imagine that a real teacher teaches a student the flow for how to solve a problem. In this scenario, we define the high-level distilled knowledge as the flow for solving a problem. As a DNN uses several sequential layers of mapping from the input space to the output space, the flow solving a problem would be defined as the relationship between features from two layers.

Most DNN with many parameters requires heavy computation for both training and testing. Those DNNs are extremely difficult to use in real-life applications as normal computers cannot handle this work. Thus, many approaches have been used to try to make networks smaller while maintaining the performance . One typical way is to distill the knowledge from the trained DNN and transfer it to a smaller network, meanwhile, a small network could be used without large storage or heavy computation. Recently, [22] introduced the model compression method based on the idea of dark knowledge. A softened version of the final output of a teacher network is used to teach information into a small student network so that the small network can learn how a large network studied given tasks in a compressed form. Net2Net [73] also uses a teacher student system with a function-preserving transformation in order to initialize the parameters of the student network, according to the teacher network. [74, 75] propose to compress the deep neural networks. [22, 76, 24] use the middle layer of a "teacher" model as the hint to guide the training of a "student" model. Additionally, knowledge distillation has a wide range of applications such as language, image, and object

detection [77, 78, 79, 23, 24, 80].

The objective of reinforcement learning (RL) is to find an optimal policy that maximizes the discounted cumulative return through the interactions with the environment [81]. The policy gradient method targets at modeling and optimizing the policy directly. The policy is usually modeled by a parameterized function such as a neural network. The value of the objective function depends on this policy and various algorithms can be applied to optimize the policy in order to maximize the rewards, for example, gradient method (REINFORCE) [82], EM algorithm [83], and natural gradient method [84].

CHAPTER 3: APPROACH

In semi-supervised ranking settings, we are given N semi-labeled training data $\{q^{(n)}, X^{(n)}, Y^{(n)}\}_{n=1}^{N}$. For each query $q^{(n)}$, $X^{(n)} = \{\mathbf{x}_1^{(n)}, \cdots, \mathbf{x}_{M_n}^{(n)}\}$ and $Y^{(n)} = \{y_1^{(n)}, \cdots, y_{M_n}^{(n)}\}$ are query-document features and relevance labels for the retrieved documents, where M_n is the number of candidate documents retrieved by query $q^{(n)}$ and y represents labels. Given a query $q^{(n)}$, our target is to retrieve a document list from $X^{(n)}$ to maximize ranking metrics, where more relevant documents are closer to the top of the ranking list.

Our co-learning consists of two modules: reinforcement ranker and classifier. We denote θ as the reinforcement ranker's parameters and ϕ as classifier's.

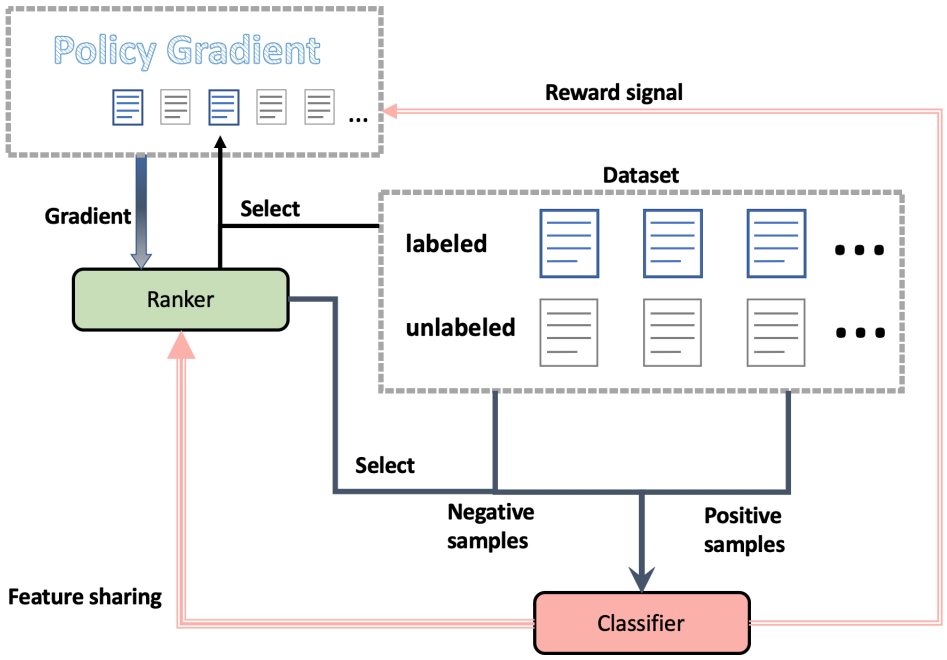

Figure 3.1: Training Pipeline contains two modules. Positive samples are from the labeled data. Negative samples are from the labeled data and drawn by the ranker. The classifier is trained by these samples. The classifier provides the reward signal and intermediate feature to the ranker. Then the ranker optimizes its ranking objective by the policy gradient.

3.1 CLASSIFIER

The classifier module is a function (could be a neural network) $f_\phi(\mathbf{x}, q)$ showed in Figure (3.2) where ϕ is the parameter, q is the query and $\mathbf{x}$ is the query-document feature. Given a query and a document, $f_\phi(\mathbf{x}, q)$ is the predicted relevance. The objective for the discriminator

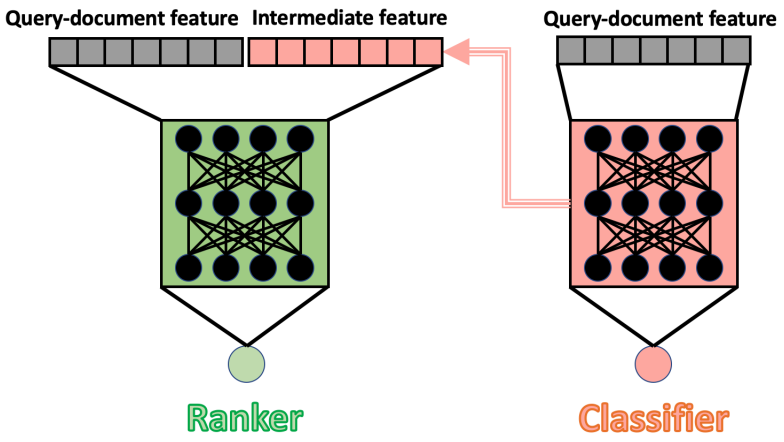

Figure 3.2: Reinforcement Ranker and Classifier are neural networks. The classifier's intermediate feature is sent to the ranker. The ranker takes query-document feature and intermediate feature as the input.

is to maximize the log-likelihood of correctly distinguishing the relevant and irrelevant document. The true relevant documents are sampled from the positively labeled training data $p_{\text{true}}(\mathbf{x} \mid q^{(n)})$, while the negative samples are generated from the negatively labeled training data and low-scoring documents given by the reinforcement ranker $p_{\theta^*}(\mathbf{x} \mid q^{(n)}, \phi^*_{\text{old}})$. We would explain $p_{\theta^*}(\mathbf{x} \mid q^{(n)}, \phi^*_{\text{old}})$ in detail referring to the Equation (3.5) later. For now, we shall know that θ^* is the current optimal reinforcement ranker and ϕ^*_{old} is the current optimal classifier.

As training is an iterative process, our new optimal ϕ^* based on θ^* and ϕ^*_{old} is obtained by

$$\begin{aligned}\phi^* = \arg\max_{\phi} \sum_{n=1}^{N} (&E_{\mathbf{x}\sim p_{\text{true}}(\mathbf{x}|q^{(n)})} \left[log(\sigma(f_\phi(\mathbf{x}, q^{(n)})))\right] + \\ &E_{\mathbf{x}\sim p_{\theta^*}(\mathbf{x}|q^{(n)}, \phi^*_{\text{old}})} \left[log(1 - \sigma(f_\phi(\mathbf{x}, q^{(n)})))\right])\end{aligned} \tag{3.1}$$

where σ is the Sigmoid function.

3.2 REINFORCEMENT RANKER

We first formalize the document ranking MDP. The definitions of state, action and transition are similar to those in MDPRank [1].

State $s_t \in S$ is the state in the ranking environment. Specifically, the agent should know the current ranking position and candidate documents. At the time step t, s_t could be represented as $\{t, X_t\}$, where the current ranking position is t and X_t comprises the

remaining documents to be retrieved.

Action $a_t \in A_t$ is to select a document as the ranking result at the step t. Since X_t is the candidate documents pool, $A_t = X_t$.

Transition $\mathcal{T} : S \times A \to S$ maps the state-action pair to a new state. After choosing a_t, we are removing document $\mathbf{x}_{(a_t)}$ from X_t, where $\mathbf{x}_{(a_t)}$ is the document chosen by a_t. Therefore

$$\mathcal{T}(\{t, X_t\}, a_t) = \{t + 1, X_t \setminus \mathbf{x}_{(a_t)}\}$$

Reward $R(s_t, a_t)$ is the immediate reward given by the environment. To optimize the quality of the ranking agent, it is natural to design the reward based on information retrieval evaluation metrics such as DCG. So we define R_{DCG}

$$R_{\text{DCG}}(s_t, a_t) = \begin{cases} 2^{y_{(a_t)}} - 1 & t = 0 \\ \frac{2^{y_{(a_t)}} - 1}{\log_2^{(t+1)}} & t > 0 \end{cases} \tag{3.2}$$

where $y_{(a_t)}$ is the relevance label of document $\mathbf{x}_{(a_t)}$. As some of the labels are unknown in the semi-supervised setting, we adopt the predicted label $\sigma(f_{\phi^*}(\mathbf{x}_{(a_t)}, q))$ from the classifier as $y_{(a_t)}$.

Following the intuition behind the classifier, in each epoch of training, ϕ^* is trained to score down the possibly irrelevant documents and score up the likely relevant documents. It is reasonable to also consider absorbing the classifier's output as the reward signal:

$$R_{\text{CLS}}(s_t, a_t) = f_{\phi^*}(\mathbf{x}_{(a_t)}, q) \tag{3.3}$$

Our final reward is a combined reward summing equation (3.2) and (3.3) together, then minus a baseline c which is a constant:

$$R(s_t, a_t) = R_{\text{DCG}}(s_t, a_t) + R_{\text{CLS}} - c. \tag{3.4}$$

Policy π_θ decides which document to choose at the current step t. It is a probabilistic distribution over all documents available. It is formally defined in Equation (3.6).

In order to share the classifier's intermediate representation to the reinforcement ranker, the classifier passes the output of its hidden layer to the reinforcement ranker. We call this sharing feature. The feature shared by the classifier ϕ^* will be concatenated with the original document-query feature $\mathbf{x}$, and together they serve as the reinforcement ranker's input. We define a probability distribution

$$p_\theta(\mathbf{x} \mid q, \phi^*) = \frac{\exp\{\mu_\theta(\mathbf{x}, s(\phi^*), q)/\Delta\}}{\sum_{\mathbf{x}\in X} \exp\{\mu_\theta(\mathbf{x}, s(\phi^*), q)/\Delta\}} \tag{3.5}$$

where $\mu_\theta(\mathbf{x}, s(\phi^*), q)$ is the reinforcement ranker function presented in Figure (3.2), $s(\phi^*)$ is the intermediate representation shared by the classifier and Δ is a constant temperature. At the step t, our candidate pool is X_t, so we derive the policy as

$$\pi_\theta(a_t \mid s_t) = \frac{\exp\{\mu_\theta(\mathbf{x}_{(a_t)}, s(\phi^*), q)/\Delta\}}{\sum_{\mathbf{x}\in X_t} \exp\{\mu_\theta(\mathbf{x}, s(\phi^*), q)/\Delta\}} \tag{3.6}$$

Policy Gradient

We train the reinforcement ranker using the policy gradient method based on the aforementioned MDP setting. The goal of the RL agent is to maximize the future expected cumulative rewards. Our designed reward is a combined reward defined in Equation (3.4). The goal is to improve the reinforcement ranker's performance by maximizing the objective:

$$\begin{aligned} \theta^* &= \arg\max_\theta U(\theta) \\ U(\theta) &= \sum_\tau P(\tau;\theta) R(\tau) \end{aligned}$$

where τ is a trajectory of state-action sequence $s_0, a_0, \cdots, s_T, a_T$ sampled by the reinforcement ranker, and $P(\tau;\theta)$ is the probability of trajectory τ under policy π_θ. $R(\tau) = \sum_{t=0}^{T} R(s_t, a_t)$. According to the policy gradient algorithm, the gradient can be calculated:

$$\begin{aligned} \nabla U(\theta) &= \nabla_\theta \sum_\tau P(\tau;\theta) R(\tau) \\ &= \sum_\tau P(\tau;\theta) \frac{\nabla_\theta P(\tau;\theta)}{P(\tau;\theta)} R(\tau) \\ &= \sum_\tau P(\tau;\theta) \nabla_\theta \log P(\tau;\theta) R(\tau) \end{aligned}$$

One can approximate this expectation by Monte-Carlo sampling. If we sample m trajectories under policy π_θ,

$$\begin{aligned}\nabla U(\theta) &\approx \frac{1}{m}\sum_{i=1}^{m}\nabla_\theta \log P(\tau^{(i)};\theta)R(\tau^{(i)}) \\ &= \frac{1}{m}\sum_{i=1}^{m}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^{(i)} \mid s_t^{(i)})R_t\end{aligned}$$

where $R_t = \sum_{k=t}^{T} R(s_k^{(i)}, a_k^{(i)})$. We further sample each time step within each trajectory so that the gradient is calculated and estimated as

$$\nabla U(\theta) \approx \nabla_\theta \log \pi_\theta(a_t \mid s_t)R_t \tag{3.7}$$

3.3 PIPELINE

The overall training pipeline is an iterative process. Its whole picture is demonstrated in Figure (3.1). Every iteration consists of two training procedures: first the classifier, then the reinforcement ranker.

When training the classifier ϕ, we keep θ^* and ϕ^*_{old} unchanged. It is because of negative sampling that training ϕ also requires θ^* and ϕ^*_{old}. To generate negative samples, we need the current reinforcement ranker θ^*, moreover, ranker needs the old classifier ϕ^*_{old} to provide it with intermediate features.

Then, we fix ϕ^* to train the reinforcement ranker θ, which first samples a bunch of trajectories. Next, the policy gradient algorithm is then used. The reward signal given by the classifier ϕ^* is passed to this reinforcement learning environment. Finally, the gradient of ϕ^* is calculated and transferred to the ranker so that the ranker can do one step of parameter update.

We summarize our reinforced co-learning method in Algorithm (3.1).

Algorithm 3.1 Reinforced Co-Learning

Input: classifier ϕ, reinforcement ranker θ, semi-labeled dataset $\{q^{(n)}, X^{(n)}, Y^{(n)}\}_{n=1}^{N}$

repeat
  $\phi^*_{\text{old}} \leftarrow \phi, \theta^* \leftarrow \theta$
  for d-steps **do**
    Use $p_\theta(\mathbf{x} \mid q, \phi^*_{\text{old}})$ to generate negative samples.
    Other samples are from labeled data.
    Update ϕ according to objective Eq. (3.1).
  end for
  $\phi^* \leftarrow \phi$
  for p-steps **do**
    Sample ranking trajectories under policy $\pi_\theta(a_t \mid s_t)$.
    Compute and accumulate rewards R_t.
    Update θ by policy gradient Eq. (3.7).
  end for
until reinforcement ranker converges

CHAPTER 4: EXPERIMENTS

4.1 DATASET

We conduct experiments on three LETOR datasets: MQ2007-semi, MQ2008-semi, and OHSUMED. The well-known LETOR is a package of benchmark datasets for research in learning-to-rank. It contains standard features, relevance judgments, data partitioning, evaluation tools, and several baselines. OHSUMED is from LETOR 3.0 while MQ2007-semi and MQ2008-semi are from LETOR 4.0. Each dataset has queries, corresponding retrieved documents, query-document features and human judged labels. Though the traditional learning-to-rank methods assume explicit relevance feedback for the query-document pair, in this paper, we focus on the real-world scenarios where more often the implicit feedback contains a huge amount of unlabeled data. In semi-supervised ranking settings, the data format is the same as that in the supervised ranking setting. The only difference is that the datasets in this setting contain both judged and unjudged query-document pairs (in the training set but not in the validation and testing set) while the dataset in supervised ranking contains only judged query-document pair.

To be noticed, the default OHSUMED dataset is not semi-supervised. We create our semi-supervised OHSUMED as it is not provided by LETOR 3.0. Our approach is to randomly label 90 percent of training data into unknown and keep the remaining 10 percent human judged labels untouched.

Label in LETOR has four types: "-1, 0, 1, 2". In this paper, we treat "1" and "2" as positive and "0" as irrelevant. Label "-1" indicates that the relevance of this query-document pair is unknown.

4.2 EVALUATION

In both MQ2007-semi and MQ2008-semi, the query-document feature is a 46-dimensional vector while OHSUMED has a 45-dimensional feature. We strictly follow the LETOR configuration, conducting a 5-fold cross-validation experiment on each dataset so that results are averaged among five data folds. On MQ2008-semi and MQ2007-semi, for each method and data fold, the validation set is used to choose the best-performed parameters. Specifically, we test the training model on a validation set to compute an NDCG@5 score for each iteration. Then we select the model with the highest NDCG@5 as the final one. For OHSUMED, we train every method until convergence and we make sure that the same amount of training

batches is used by different methods. Finally, the mean of 5-fold testing results is reported in this paper as the final score. Our code base, as well as training profile, will be released on Github.

Discounted cumulative gain (DCG) is a measure of ranking quality. It is accumulated from the top results to the bottom, aiming to measure the gain or relevance of the result list based on positions. The formulation of DCG at the particular position p is defined

$$\mathrm{DCG}_p = \sum_{i=1}^{p} \frac{\mathrm{rel}_i}{\log_2(i+1)}$$

Search results vary depending on different queries. To make consistent comparison, Normalized DCG (NDCG) is used in this paper

$$\mathrm{NDCG}_p = \frac{\mathrm{DCG}_p}{\mathrm{IDCG}_p}$$

where IDCG is ideal DCG. This is done by sorting the retrieved document list by relevance score, producing the possible maximum DCG through p.

Next, we use Precision@k to measure the proportion of retrieved documents in the top-k list that are relevant:

$$\text{Precision@}k = \frac{\text{number of relevant documents in top-k}}{k}$$

Finally, for each data fold, mean average precision (MAP) is calculated to measure the average precision scores for all queries.

$$\mathrm{MAP} = \frac{\sum_{q=1}^{Q} \mathrm{AveP}(q)}{Q}$$
$$\mathrm{AveP}(q) = \frac{\sum_{k=1}^{n} (\text{Precision@}k \times \mathrm{rel}(k))}{\text{number of relevant documents}}$$

where Q is the number of queries and $\mathrm{rel}(k)$ is a 0-1 indicator.

4.3 COMPARED METHODS

We first compare our method with MDPRank. Both MDPRank and co-learning consider the ranking process as an MDP and use ranking metric NDCG as the reward. The apparent difference is that MDPRank is designed as a supervised method and relies on explicit relevance feedback, which is not possible in semi-supervised environments. In our experiments,

	P@1	p@3	p@5	p@10	MAP
MDPRank [1]	0.5620	0.4972	0.4593	0.3968	0.6306
IRGAN [4]	0.5468	0.4816	0.4415	0.3847	0.6105
Co-learning	**0.6225**	**0.5300**	**0.4701**	**0.4039**	**0.6580**
	NDCG@1	NDCG@3	NDCG@5	NDCG@10	
MDPRank [1]	0.5620	0.5886	0.6458	0.6144	
IRGAN [4]	0.5468	0.5686	0.6197	0.5873	
Co-learning	**0.6225**	**0.6327**	**0.6751**	**0.6355**	

Table 4.1: MQ2008-Semi

	P@1	p@3	p@5	p@10	MAP
MDPRank [1]	0.4976	0.4630	0.4346	0.4021	0.4971
IRGAN [4]	0.5073	0.4768	0.4523	0.4165	0.5113
Co-learning	**0.5190**	**0.4839**	**0.4638**	**0.4266**	**0.5233**
	NDCG@1	NDCG@3	NDCG@5	NDCG@10	
MDPRank [1]	0.4976	0.4861	0.4786	0.4969	
IRGAN [4]	0.5073	0.5011	0.4973	0.5145	
Co-learning	**0.5190**	**0.5093**	**0.5097**	**0.5276**	

Table 4.2: MQ2007-Semi

the semi-supervised training dataset is divided into two parts: human judged and unlabeled data. We only use human judged training data to train MDPRank. At the inference time, the same testing data is used for MDPRank and co-learning. From this comparison, we show how much performance gain that co-learning could accomplish by using the semi-supervised data.

Secondly, we compare our method with the state-of-art semi-supervised algorithm IRGAN. IRGAN proposed a min-max game model to combine the two perspectives of thinking in information retrieval: the generative model and the discriminative model. The generative model tries to generate relevant documents, given a query, while the discriminative model learns to discriminate well-matched documents from the ill-matched ones by predicting a relevancy.

In the semi-supervised ranking tasks, the objective for the generator is given a query to select documents that are most likely to be relevant and thus could fool the discriminator. The discriminator constantly learns to distinguish between generated relevant documents and ground truths. The whole model is trained iteratively: at each iteration, fake samples that the generator draws from semi-supervised data, are combined with ground truth samples, serving as the discriminator's training data. The policy gradient is used to maximize the generator's ability to select more relevant samples from candidates. This algorithm finally

	P@1	p@3	p@5	p@10	MAP
MDPRank [1]	0.4545	0.4545	0.4454	0.3545	0.3312
IRGAN [4]	**0.5000**	0.4393	0.4272	0.3681	0.3436
Co-learning	**0.5000**	**0.4848**	**0.4454**	**0.3727**	**0.3511**
	NDCG@1	NDCG@3	NDCG@5	NDCG@10	
MDPRank [1]	0.4545	0.4517	0.4547	0.4118	
IRGAN [4]	**0.4999**	0.4629	0.4583	0.4319	
Co-learning	**0.4999**	**0.5028**	**0.4745**	**0.4380**	

Table 4.3: OHSUMED

evaluates the generator's ranking performance on the testing dataset. Given a query, the generator first calculates a softmax score for each document, then sorts these documents based on scores as the ranking list.

One might think of the classifier in co-learning as the discriminator's counterpart and the reinforcement ranker as the generator's counterpart. Perhaps the biggest difference between IRGAN and co-learning is the training objective. IRGAN is playing a generative adversarial game by training an overall min-max objective to iteratively push the generator, generating more relevant fake samples and the discriminator to draw a clearer boundary between real and fake samples. Meanwhile, the co-learning is also trained iteratively by two separate modules, but with different objectives. The classifier is an ordinary classification model with cross-entropy loss. The reinforcement ranker treats ranking as an MDP and the policy gradient algorithm is used to optimize the ranking metric (NDCG).

Despite both the co-learning and IRGAN use the policy gradient, the ranking metric (NDCG) as part of combined rewards, helps our method obtain better performance, particularly in OHSUMED. Another insight by co-learning that makes a huge difference is the knowledge distillation structure. By combining these novel techniques together, our co-learning method accomplishes considerable improvement over baselines and offers a very robust learning algorithm for semi-supervised ranking tasks.

4.4 PERFORMANCE

As shown in Tables (4.1), (4.2), and (4.3), we provide the overall performance of all the baseline algorithms on the MQ2008-semi, MQ2007-semi and OHSUMED datasets. In our co-learning framework, we use the reinforcement ranker to predict the user-preferred document ranking list, given a query. This is done by the RL agent performing the softmax sampling with the temperature parameter set very close to 0. From the tables, we observe that the co-learning constantly achieves the highest score in terms of all ranking metrics.

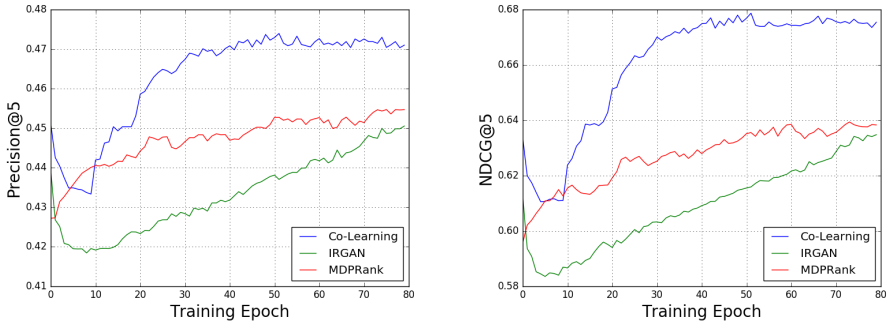

Figure 4.1: Learning Curves on MQ2008-semi

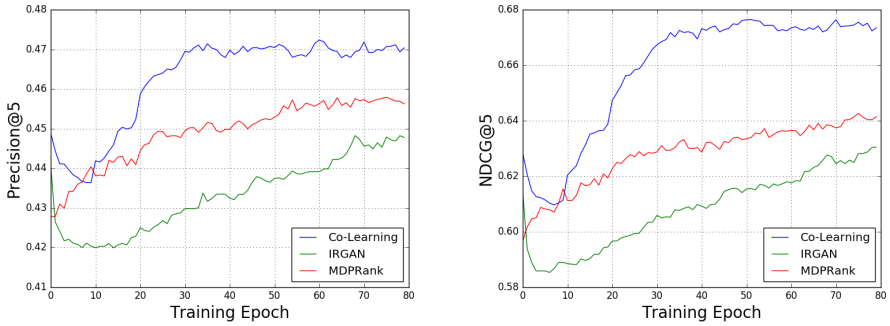

Figure 4.2: Testing Curves on MQ2008-semi

Next, we take a deeper look at the learning and testing curves. As mentioned before, since our final model is selected by the NDCG@5 score from training iterations, we will also illustrate the learning and testing in terms of the NDCG@5, where learning curves are tested on the validation set while testing curves are tested on the testing dataset.

On MQ2008-semi, co-learning improves drastically on P@1 and NDCG@1, which measure the quality of the very top document in the ranking list. It should be noted that MDPRank actually works better than IRGAN. Figure (4.1) and (4.2) show that a simple model like MDPRank could generalize very well, and achieve considerable performance. Even if only trained on labeled data, MDPRank dramatically beats IRGAN at the early stage. As training goes by, IRGAN approaches MDPRank but still is outperformed until convergence. This observation suggests that a simple and robust method will likely outperform over a complex model that uses semi-supervised data. On the contrary, co-learning demonstrates a very strong generalization power over baselines. Our method constantly pushes the limit in the whole training procedure with similar structural and parameter complexities as IRGAN.

On MQ2007-semi, co-learning again shows its superiority over baselines. As Figure (4.3) and (4.4) display, both semi-supervised methods outperform MDPRank. This time, co-

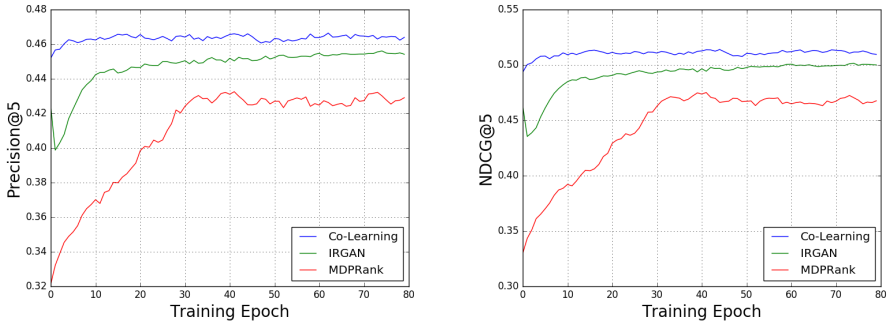

Figure 4.3: Learning Curves on MQ2007-semi

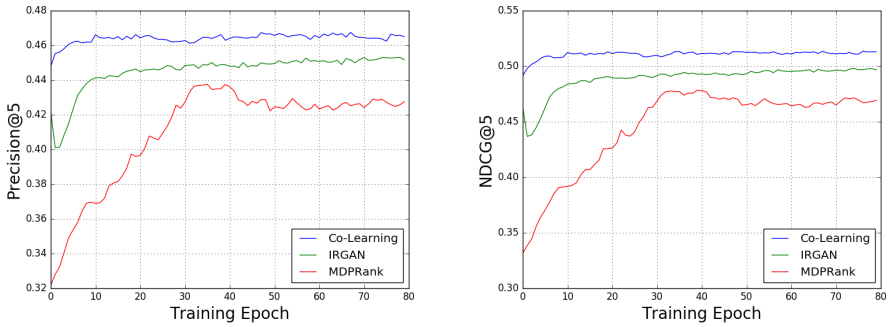

Figure 4.4: Testing Curves on MQ2007-semi

learning converges to a very high performance after only few iterations, then keeps maintaining the top performance as training lasts. It demonstrates that co-learning not only contains higher generalization power, but also good convergence and robustness properties. Finally, OHSUMED is a small dataset compared with the aforementioned ones. Our method once again pushes the limit and shows robustness (Figure (4.5), (4.6)).

4.5 ABLATION STUDY

The significant performance improvement of reinforced Co-Learning is due primarily to the combined reward and knowledge distillation. To elucidate the contributions of different techniques, we conduct ablation studies in this section. In order to obtain a more thorough understanding of co-learning, we propose three reductions of co-learning:

1. Co-learning without knowledge distillation
2. Co-learning without combined reward
3. Co-learning without iterative training

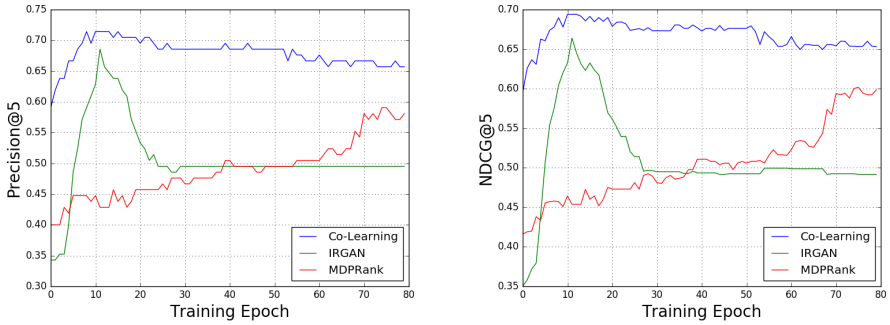

Figure 4.5: Learning Curves on OHSUMED

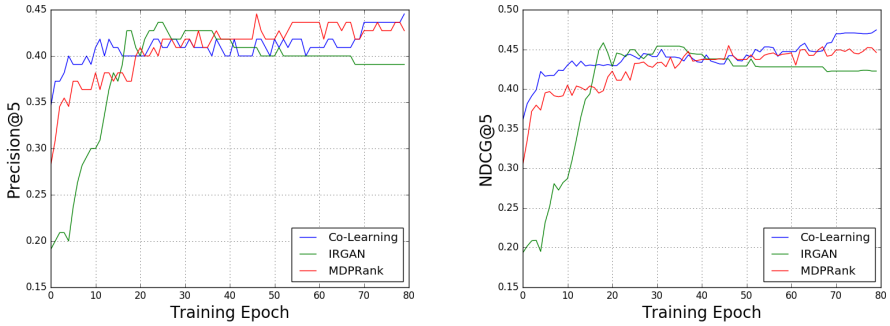

Figure 4.6: Testing Curves on OHSUMED

In the method (1), we want to evaluate the contribution of knowledge distillation. So we turn off the feature sharing path between classifier and reinforcement ranker. Method (2) uses R_{CLS} instead of combined reward. Method (3) trains the classifier first, then fixes the classifier to train the reinforcement ranker. Throughout the process, the overall amount of training data remains the same.

Each of these methods only reduces one particular function from the co-learning and keeps everything else unchanged. Our ablation study is to observe these methods' performance. The lower performance of one method means its counterpart function contributes more significantly. Additionally, we define a relative improvement (RI) ratio for each ranking metric M:

$$\text{RI}_M = \frac{M_{\text{Co-Learning}} - M_{\text{Co-Learning with Reduction}}}{M_{\text{Co-Learning}}} \times 100\%$$

Table (4.5) shows that knowledge distillation contributes significantly in MQ2008-semi. When it refers to some metrics P@1, p@3, NDCG@1 and NDCG@3, relative improvement could reach nearly 10%. Iterative training also contributes about 3% in these metrics. In MQ2007-semi, similar results are listed in Table (4.6), where knowledge distillation contributes more than 1% in most metrics. However, it is not always the case that knowledge

distillation contributes the most . Table (4.7) emphasizes that combined reward could also play an important role in the co-learning framework. In OHSUMED, the combined reward has achieved incredible relative improvement: 10% P@1. 12.50% P@3, 8.16% P@5 and 8.54% P@10.

In conclusion, our ablation study demonstrates that knowledge distillation, combined reward, and iterative training all contribute proportionally in the co-learning framework. It is for this reason, that we design our co-learning method using these aforementioned techniques.

4.6 EXPERIMENT SETTING

The classifier and reinforcement ranker are three-layer neural networks, including the input and output. Their hidden layers' sizes are designed in the same way as the query-document feature size. The classifier takes the query-document features as input. Query-document features are 46 dimensional in MQ2008-semi, MQ2007-semi and 45 in OHSUMED. Since we add the feature sharing layer to distill the intermediate representations from the classifier to ranking RL, the RL network's input is query-document feature plus the shared representation. The bias c in the combined reward Equation (3.4) is 0.5 and emperature Δ in Equation (3.6) is set to 0.2.

Parameters	Setting
Classifier input dims	46 (45 in OHSUMED)
Classifier hidden layer dims	46 (45 in OHSUMED)
Classifier training batch size	8
Classifier learning rate	0.001
RL input dims	92 (90 in OHSUMED)
RL hidden layer dims	46 (45 in OHSUMED)
RL learning rate	0.001
RL reward bias	0.5
Temperature	0.2

Table 4.4: Parameter settings

	P@1	p@3	p@5	p@10	MAP
CWKD	0.5550	0.4811	0.4408	0.3843	0.6120
KDI	10.84%	9.23%	6.23%	4.85%	6.99%
CWCR	0.6043	0.5278	0.4689	0.4003	0.6559
CRI	2.92%	0.42%	0.26%	0.89%	0.32%
CWIT	0.6016	0.5154	0.4663	0.4037	0.6502
ITI	3.35%	2.75%	0.81%	0.04%	1.18%
CL	0.6225	0.5300	0.4701	0.4039	0.6580
	NDCG@1	NDCG@3	NDCG@5	NDCG@10	
CWKD	0.5550	0.5699	0.6212	0.5904	
KDI	10.84%	9.93%	7.98%	4.51%	
CWCR	0.6043	0.6295	0.6717	0.6311	
CRI	2.92%	0.51%	0.50%	0.69%	
CWIT	0.6016	0.6167	0.6646	0.6306	
ITI	3.35%	2.52%	1.54%	0.77%	
CL	0.6225	0.6327	0.6751	0.6355	

Table 4.5: MQ2008-Semi Ablation Study, where CWKD represents Co-learning Without Knowledge Distillation, KDI represents Knowledge Distillation Improvement, CWCR represents Co-learning Without Combined Reward, CRI represents Combined Reward Improvement, CWIT represents Co-learning Without Iterative Training, ITI represents Iterative Training Improvement, and CL represents Co-learning

	P@1	p@3	p@5	p@10	MAP
CWKD	0.5074	0.4796	0.4547	0.4180	0.5133
KDI	1.20%	0.51%	1.58%	1.06%	1.46%
CWCR	0.5135	0.4821	0.4620	0.4224	0.5208
CRI	1.06%	0.37%	0.38%	0.97%	0.47%
CWIT	0.5252	0.4807	0.4619	0.4250	0.5227
ITI	-1.20%	0.66%	0.41%	0.36%	0.16%
CL	0.5190	0.4839	0.4638	0.4266	0.5234
	NDCG@1	NDCG@3	NDCG@5	NDCG@10	
CWKD	0.5074	0.5033	0.5002	0.5172	
KDI	1.20%	0.74%	1.49%	1.22%	
CWCR	0.51352	0.5070	0.5077	0.5235	
CRI	1.06%	0.44%	0.38%	0.77%	
CWIT	0.5252	0.5086	0.5092	0.5266	
ITI	-1.20%	0.14%	0.09%	0.20%	
CL	0.5190	0.5093	0.5097	0.5276	

Table 4.6: MQ2007-Semi Ablation Study, where CWKD represents Co-learning Without Knowledge Distillation, KDI represents Knowledge Distillation Improvement, CWCR represents Co-learning Without Combined Reward, CRI represents Combined Reward Improvement, CWIT represents Co-learning Without Iterative Training, ITI represents Iterative Training Improvement, and CL represents Co-learning

	P@1	p@3	p@5	p@10	MAP
CWKD	0.5000	0.4696	0.4363	0.3636	0.3461
KDI	0.00%	3.13%	2.04%	2.44%	1.41%
CWCR	0.4500	0.4242	0.4090	0.3409	0.3400
CRI	10.00%	12.50%	8.16%	8.54%	3.15%
CWIT	0.5000	0.4697	0.4272	0.3727	0.3474
ITI	0.00%	3.13%	4.08%	0.00%	1.046%
CL	0.5000	0.4848	0.4455	0.3727	0.3511
	NDCG@1	NDCG@3	NDCG@5	NDCG@10	
CWKD	0.5000	0.4893	0.4700	0.4327	
KDI	0.00%	2.68%	0.94%	1.20%	
CWCR	0.4500	0.4680	0.4553	0.4131	
CRI	10.00%	6.92%	4.04%	5.69%	
CWIT	0.5000	0.4787	0.4518	0.4286	
ITI	0.00%	4.80%	4.79%	2.15%	
CL	0.5000	0.5028	0.4745	0.4380	

Table 4.7: OHSUMED Ablation Study, where CWKD represents Co-learning Without Knowledge Distillation, KDI represents Knowledge Distillation Improvement, CWCR represents Co-learning Without Combined Reward, CRI represents Combined Reward Improvement, CWIT represents Co-learning Without Iterative Training, ITI represents Iterative Training Improvement, and CL represents Co-learning

CHAPTER 5: DISCUSSION AND CONCLUSION

5.1 DISCUSSION

In this paper, we convert different relevance levels into positive $\{1, 2\}$ and negative $\{0\}$ similar to IRGAN. In LETOR datasets, relevance label has four types $\{0, 1, 2, -1\}$. 1 and 2 are two levels of relevance. 0 means irrelevant and -1 is unlabeled data meaning unknown. Using a binary label setting is based on the observation that most positive relevant documents are 1 in LETOR datasets (about 70%). However, this setting would lead to a smaller IDCG, as label 2 is converted to 1. As a result, the reported NDCG scores would scale up.

We empirically compare other traditional learning-to-rank methods with baselines. The performance of MDPRank and some other learning to rank methods are listed in MQ2007, Table (5.1). The scale of reported numbers in different papers differs because of different experiment settings. Different from the standard LETOR 5-fold setting, IRGAN uses another data fold schema, so the results are also different from those in the 5-fold setting. Its experimental results are referred to MQ2008-semi in Table (5.1).

MQ2007:

	NDCG@3	NDCG@5	NDCG@10
ListNet	0.4091	0.4170	0.4440
AdaRank	0.4044	0.4102	0.4369
SVMMAP	0.3899	0.3983	0.4187
MDPRank	0.4101	0.4147	0.4416

MQ2008-semi:

	NDCG@3	NDCG@5	NDCG@10
RankNet	0.1801	0.1709	0.1943
LambdaRank	0.1926	0.1920	0.2093
LambdaMART	0.1573	0.1456	0.1627
IRGAN	0.2065	0.2225	0.2483

Table 5.1: Other baselines

5.2 CONCLUSION

In this paper, we have proposed a reinforced co-learning framework for semi-supervised ranking tasks. Our co-learning leverages the learning of a large amount of unlabeled data and the direct optimization of ranking scores at the same time.

The classifier module and reinforcement ranker module are optimized iteratively using semi-labeled data. The classifier gives each document a predicted relevance score while the reinforcement ranker directly optimizes the ranking measures using the policy gradient where the reward signal in this MDP is given by the classifier module. A combined reward is designed and given to the reinforcement ranker. A feature sharing path connecting two modules distills the classifier's intermediate features to the learning of the reinforcement ranker. Experimental results based on three LETOR datasets show that reinforced co-learning outperforms state-of-art ranking methods. In addition, extensive ablation studies are conducted to deeply understand each function in the co-learning framework .

REFERENCES

[1] Z. Wei, J. Xu, Y. Lan, J. Guo, and X. Cheng, "Reinforcement learning to rank with markov decision process," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2017, pp. 945–948.

[2] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems*, 2000, pp. 1057–1063.

[3] O. Chapelle, B. Scholkopf, and A. Zien, "Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]," *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 542–542, 2009.

[4] J. Wang, L. Yu, W. Zhang, Y. Gong, Y. Xu, B. Wang, P. Zhang, and D. Zhang, "Irgan: A minimax game for unifying generative and discriminative information retrieval models," in *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 2017, pp. 515–524.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[6] Y. LeCun, Y. Bengio, and G. E. Hinton, "Deep learning," *Nature*, 2015.

[7] A. Krizhevsky, I. Sutskever, , and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012.

[8] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. NIPS*, 2014.

[9] G. E. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, 2012.

[10] Y. Bengio, A. Courville, and P. Vincent, "Representation Learning: A Review and New Perspectives," *PAMI*, 2013.

[11] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[12] W. P. Powell, *Approximate Dynamic Programming*. Wiley, 2011.

[13] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[14] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *JMLR*, 1996.

[15] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, University of Cambridge England, 1989.

[16] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, 1992.

[17] J. N. Tsitsiklis and B. V. Roy, "An analysis of temporal-difference learning with function approximation," 1997.

[18] M. Riedmiller, "Neural fitted Q iteration - first experiences with a data efficient neural reinforcement learning method," in *Proc. ECML*, 2005.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," in *NIPS Deep Learning Workshop*, 2013.

[20] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, 2015.

[21] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, 2016.

[22] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[23] Y. Gong, L. Wang, M. Hodosh, J. Hockenmaier, and S. Lazebnik, "Improving image-sentence embeddings using large weakly annotated photo collections," in *European Conference on Computer Vision*. Springer, 2014, pp. 529–545.

[24] G. Chen, W. Choi, X. Yu, T. Han, and M. Chandraker, "Learning efficient object detection models with knowledge distillation," in *Advances in Neural Information Processing Systems*, 2017, pp. 742–751.

[25] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, "Learning to rank using gradient descent," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 89–96.

[26] R. Baeza-Yates, B. Ribeiro-Neto et al., *Modern information retrieval*. ACM press New York, 1999, vol. 463.

[27] E. F. Harrington, "Online ranking/collaborative filtering using the perceptron algorithm," in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 2003, pp. 250–257.

[28] M. Collins, "Ranking algorithms for named-entity extraction: Boosting and the voted perceptron," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2002, pp. 489–496.

[29] J. Xu, Y. Cao, H. Li, and M. Zhao, "Ranking definitions with supervised learning methods," in *Special interest tracks and posters of the 14th international conference on World Wide Web*. ACM, 2005, pp. 811–819.

[30] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," in *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2005, pp. 115–124.

[31] K. Dave, S. Lawrence, and D. M. Pennock, "Mining the peanut gallery: Opinion extraction and semantic classification of product reviews," in *Proceedings of the 12th international conference on World Wide Web*. ACM, 2003, pp. 519–528.

[32] W. Chu and Z. Ghahramani, "Gaussian processes for ordinal regression," *Journal of machine learning research*, vol. 6, no. Jul, pp. 1019–1041, 2005.

[33] W. S. Cooper, F. C. Gey, and D. P. Dabney, "Probabilistic retrieval based on staged logistic regression," in *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 1992, pp. 198–210.

[34] D. Cossock and T. Zhang, "Subset ranking using regression," in *International Conference on Computational Learning Theory*. Springer, 2006, pp. 605–619.

[35] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon, "Adapting ranking svm to document retrieval," in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2006, pp. 186–193.

[36] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer, "An efficient boosting algorithm for combining preferences," *Journal of machine learning research*, vol. 4, no. Nov, pp. 933–969, 2003.

[37] T. Qin, X.-D. Zhang, D.-S. Wang, T.-Y. Liu, W. Lai, and H. Li, "Ranking with multiple hyperplanes," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2007, pp. 279–286.

[38] C. J. Burges, R. Ragno, and Q. V. Le, "Learning to rank with nonsmooth cost functions," in *Advances in neural information processing systems*, 2007, pp. 193–200.

[39] T. Qin, T.-Y. Liu, M.-F. Tsai, X.-D. Zhang, and H. Li, "Learning to search web pages with query-level loss functions," *Technical Report*, vol. 156, 2006.

[40] M. Taylor, J. Guiver, S. Robertson, and T. Minka, "Softrank: optimizing non-smooth rank metrics," in *Proceedings of the 2008 International Conference on Web Search and Data Mining*. ACM, 2008, pp. 77–86.

[41] F. Xia, T.-Y. Liu, J. Wang, W. Zhang, and H. Li, "Listwise approach to learning to rank: theory and algorithm," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1192–1199.

[42] Y. Yue, T. Finley, F. Radlinski, and T. Joachims, "A support vector method for optimizing average precision," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2007, pp. 271–278.

[43] C. J. Burges, "From ranknet to lambdarank to lambdamart: An overview," *Learning*, vol. 11, no. 23-581, p. 81, 2010.

[44] S. Chakrabarti, R. Khanna, U. Sawant, and C. Bhattacharyya, "Structured learning for non-smooth ranking losses," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 88–96.

[45] J. Xu and H. Li, "Adarank: a boosting algorithm for information retrieval," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2007, pp. 391–398.

[46] J. Xu, T.-Y. Liu, M. Lu, H. Li, and W.-Y. Ma, "Directly optimizing evaluation measures in learning to rank," in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2008, pp. 107–114.

[47] O. Chapelle and M. Wu, "Gradient descent optimization of smoothed information retrieval metrics," *Information retrieval*, vol. 13, no. 3, pp. 216–235, 2010.

[48] X. Zhu, "Semi-supervised learning literature survey," *Computer Science, University of Wisconsin-Madison*, vol. 2, no. 3, p. 4, 2006.

[49] D. Yarowsky, "Unsupervised word sense disambiguation rivaling supervised methods," in *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 1995, pp. 189–196.

[50] S. Abney, "Understanding the yarowsky algorithm," *Computational Linguistics*, vol. 30, no. 3, pp. 365–395, 2004.

[51] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proceedings of the eleventh annual conference on Computational learning theory*. ACM, 1998, pp. 92–100.

[52] V. Castelli and T. M. Cover, "The relative value of labeled and unlabeled samples in pattern recognition with an unknown mixing parameter," *IEEE Transactions on information theory*, vol. 42, no. 6, pp. 2102–2117, 1996.

[53] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell, "Text classification from labeled and unlabeled documents using em," *Machine learning*, vol. 39, no. 2-3, pp. 103–134, 2000.

[54] E. Riloff, J. Wiebe, and T. Wilson, "Learning subjective nouns using extraction pattern bootstrapping," in *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*. Association for Computational Linguistics, 2003, pp. 25–32.

[55] B. Maeireizo, D. Litman, and R. Hwa, "Co-training for predicting emotions with spoken dialogue data," in *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*. Association for Computational Linguistics, 2004, p. 28.

[56] C. Rosenberg, M. Hebert, and H. Schneiderman, "Semi-supervised self-training of object detection models." in *WACV/MOTION*, 2005, pp. 29–36.

[57] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1977.

[58] K. Nigam and R. Ghani, "Analyzing the effectiveness and applicability of co-training," in *Proceedings of the ninth international conference on Information and knowledge management*. ACM, 2000, pp. 86–93.

[59] M. Collins and Y. Singer, "Unsupervised models for named entity classification," in *1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 1999.

[60] R. Jones, "Learning to extract entities from labeled and unlabeled text," Ph.D. dissertation, Citeseer, 2005.

[61] M.-F. Balcan and A. Blum, "21 an augmented pac model for semi-supervised learning," 2006.

[62] Z.-H. Zhou, D.-C. Zhan, and Q. Yang, "Semi-supervised learning with very few labeled training examples," in *AAAI*, 2007, pp. 675–680.

[63] J. Wu, L. Li, and W. Y. Wang, "Reinforced co-training," *arXiv preprint arXiv:1804.06035*, 2018.

[64] D. Maclaurin, D. Duvenaud, and R. Adams, "Gradient-based hyperparameter optimization through reversible learning," in *International Conference on Machine Learning*, 2015, pp. 2113–2122.

[65] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.

[66] Y. Chen, M. W. Hoffman, S. G. Colmenarejo, M. Denil, T. P. Lillicrap, M. Botvinick, and N. de Freitas, "Learning to learn without gradient descent by gradient descent," *arXiv preprint arXiv:1611.03824*, 2016.

[67] O. Wichrowska, N. Maheswaranathan, M. W. Hoffman, S. G. Colmenarejo, M. Denil, N. de Freitas, and J. Sohl-Dickstein, "Learned optimizers that scale and generalize," *arXiv preprint arXiv:1703.04813*, 2017.

[68] V. Sindhwani, P. Niyogi, and M. Belkin, "Beyond the point cloud: from transductive to semi-supervised learning," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 824–831.

[69] U. Brefeld and T. Scheffer, "Semi-supervised learning for structured output variables," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 145–152.

[70] U. Brefeld, C. Büscher, and T. Scheffer, "Multi-view discriminative sequential learning," in *European Conference on Machine Learning*. Springer, 2005, pp. 60–71.

[71] B. Leskes, "The value of agreement, a new boosting algorithm," in *International Conference on Computational Learning Theory*. Springer, 2005, pp. 95–110.

[72] J. Farquhar, D. Hardoon, H. Meng, J. S. Shawe-taylor, and S. Szedmak, "Two view learning: Svm-2k, theory and practice," in *Advances in neural information processing systems*, 2006, pp. 355–362.

[73] T. Chen, I. Goodfellow, and J. Shlens, "Net2net: Accelerating learning via knowledge transfer," *arXiv preprint arXiv:1511.05641*, 2015.

[74] C. Bucilu, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006, pp. 535–541.

[75] J. Ba and R. Caruana, "Do deep nets really need to be deep?" in *Advances in neural information processing systems*, 2014, pp. 2654–2662.

[76] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," *arXiv preprint arXiv:1412.6550*, 2014.

[77] S. Gupta, J. Hoffman, and J. Malik, "Cross modal distillation for supervision transfer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2827–2836.

[78] J.-C. Su and S. Maji, "Cross quality distillation." *arXiv preprint arXiv:1604.00433*, 2016.

[79] J. Shen, N. Vesdapunt, V. N. Boddeti, and K. M. Kitani, "In teacher we trust: Learning compressed models for pedestrian detection," *arXiv preprint arXiv:1612.00478*, 2016.

[80] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[81] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[82] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[83] P. Dayan and G. E. Hinton, "Using expectation-maximization for reinforcement learning," *Neural Computation*, vol. 9, no. 2, pp. 271–278, 1997.

[84] S. M. Kakade, "A natural policy gradient," in *Advances in neural information processing systems*, 2002, pp. 1531–1538.